Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 28 (2014) 505 – 512

Procedia Computer Science

Conference on Systems Engineering Research (CSER 2014)

Eds.: Azad M. Madni, University of Southern California; Barry Boehm, University of Southern California; Michael Sievers, Jet Propulsion Laboratory; Marilee Wheaton, The Aerospace Corporation
Redondo Beach, CA, March 21-22, 2014

On Technical Credit

Brian Berenbach

INCOSE, ESEP Member, 7670 Opportunity Rd., Ste. 220, San Diego CA 92111 USA

Abstract

"Technical Debt" is a term first used by Ward Cunningham in an experience report in 1992.[1] The term refers to the accruing debt or downstream cost that happens when short term priorities trump long term lifecycle costs. The term, when introduced, was used in the context of the development of software systems. However, since 1992, the field of systems engineering has evolved, and it has been found that technical debt also applies to the development and construction of systems. This paper takes a contrary view; technical debt is discussed mostly in the context of bad practices; the author contends that the focus should be on system principles that preclude the introduction, either anticipated or unanticipated, of negative lifecycle impacts. A set of heuristics is presented that describes what *should* be done rather than what should *not* be done. From these heuristics, some emergent trends will be identified. Such trends may be leveraged to design systems with reduced long term lifecycle costs and, on occasion, unexpected benefits.

Keywords: Technical Debt, Technical Credit, lifecycle costs, systems engineering

1. Introduction

The term "Technical Debt" was first introduced by Ward Cunningham in an OOPSLA'92 experience report.[1] In that report, he described the early shipment or delivery of incomplete or insufficiently tested software as "going into debt". That is, the debt would have to be repaid later with interest, as the cost of repairing a delivered system is known to be considerably more expensive than the initial in-house cost to construct and fully test the complete system in the first place.[2] Since then, the definition has been expanded to include the debt incurred by an enterprise[3],

doi:10.1016/j.procs.2014.03.062

and there have been whole workshops dedicated to the subject.[4] While the accruing of technical debt is generally considered to be the result of poor engineering practices, practitioners have proposed cleaner definitions.[5] The flip side of technical debt is *technical credit*. Other than a blog by Neil Bornstein[6], I have been unable to find any credible research or publications on the topic. Consequently, I am proposing a definition that is inclusive of both software and systems engineering:

Technical Credit is the investment in the engineering, designing and constructing of software or systems over and above the minimum necessary effort, in anticipation of emergent properties paying dividends at a later date.

I suggest certain heuristics to enable the long term benefit of such an investment:

- Risk-centric analysis
- Error-centric design
- Macro and micro extensibility
- Adaptive and autonomous principles applied, where possible, to subsystems and components
- Over-engineering wherever possible, consistent with resources and schedule
- Market-centric approach to requirements for subsystems and components

The heuristics are further described in the paragraphs below.

Corollary: Unforeseen or unanticipated emergent properties may result over time providing a significant return on investment; in many cases also providing unforeseen benefits to the manufacturer and/or to society at large.

Where technical debt describes the results of deliberate or unintended poor practices, technical credit accrues when there is intentional over-engineering (not gold-plating) of a product or system. In order for the reader to better understand the benefits of technical credit, I will describe what I believe to be an example of a product built using some, if not all, of the heuristics of technical credit.

The Ponte Vecchio (see figure 1) is a medieval stone bridge over the river Arno in Italy. After having been destroyed several times by flood, it was rebuilt in 1345.[7] Since then, it has survived floods and World War II. As the science of bridge building was fraught with uncertainty at the time of construction, the builders relied on over-engineering (e.g. the use of stone instead of wood). In 1966 there was a catastrophic flood of the Arno that destroyed many historical relics in Florence.[8] However, the Ponte Vecchio was left unscathed. Contrast that with a 2013 flood in Colorado, USA that destroyed 30 bridges.[9] So on the one hand we have a bridge that has not needed rebuilding since 1345, and on the other we have 30 bridges in one US state that needed rebuilding after one flood. While it is not feasible to calculate an ROI for the Ponte Vecchio, it is clear that its longevity has resulted in considerable savings, both from not requiring reconstruction, and from continuous tourist revenue.

1.1. Emergent properties of the Ponte Vecchio

The Ponte Vecchio has become one of the most important tourist attractions in Florence, bringing in millions of dollars in revenue each year to the city.[10] The bridge was built in 1345; tourism was not something that was even conceived of by the designers. Hence, it is an emergent property. The Corridoio Vasariano is a private passageway built on the top of the Ponte Vecchio connecting private residences on both sides of the Arno. The Vasari Corridor was built by Grand Duke Cosimo I de' Medici in 1564 so that his son, Francesco, could visit his wife, Johanna of Austria, without having to walk among the “riff-raff” in the streets, e.g. it was built for love.[11] Eventually, the tunnel was turned into an art gallery that, today, requires special permission to visit. It is most unlikely that any of the emergent properties (tourism, tunnel of love, art gallery) of the Ponte Vecchio were foreseen by the original builders!

The sections below describe heuristics for investing in technical credit above and beyond traditional analysis, design and construction techniques that may yield unforeseen benefits through emerging properties, as well as improved traditional benefits thru increased reliability, safety and longevity.

Figure 1.Ponte Vecchio Bridge

2. Risk-centric Analysis

During the early phases of a project, risk and hazard analysis are typically addressed as follows:

- Requirements are analyzed for potential hazards. Depending on the regulatory environment, a hazard analysis is performed for each requirement for which there may be a risk of injury (personal) or damage (property). If the risk is deemed significant (risk is a function of severity and probability of occurrence), a hazard analysis is completed. Depending on the results, mitigating procedures or requirements may be added to the requirement set.
- After a preliminary design is complete various safety analyses are performed, such as failure mode effects analyses (FMEA).[12] Designs are modified to eliminate any discovered hazards.

There are many additional analyses that may be performed, some of them domain specific. For example, in the nuclear industry, software programs or codes are utilized to demonstrate the safety of designs, e.g. equipment sizing, heat removal analysis, etc. However, risk assessment is normally done in reaction to potential hazards noted by the customer or expert, or those found through analyses. . Furthermore, these analyses tend to be stovepipe in nature, looking at a part of the system, without looking holistically at the entire environment, processes and lifecycle. De Neufville has studied applying system dynamics to forest fire management, extending the analysis beyond pure scientific analysis to include the Portuguese political system.[13] Nancy Leveson, in her text *Engineering a Safer World* [14], does a thorough analysis of the Bhopal disaster in 1984. A chain of events, including human and mechanical failure resulted in 2000 fatalities and over 200,000 injuries. In January of 2000 an Alaskan Airlines MD-80 crashed killing all 88 people on board. The crash was tied to failure to periodically lubricate the jack screw that controlled the horizontal tail position.[15] Risk-centric analysis refers to the analysis of the complete environment of the system under construction, with extended causal analyses and removal of potential “operator error” by shifting

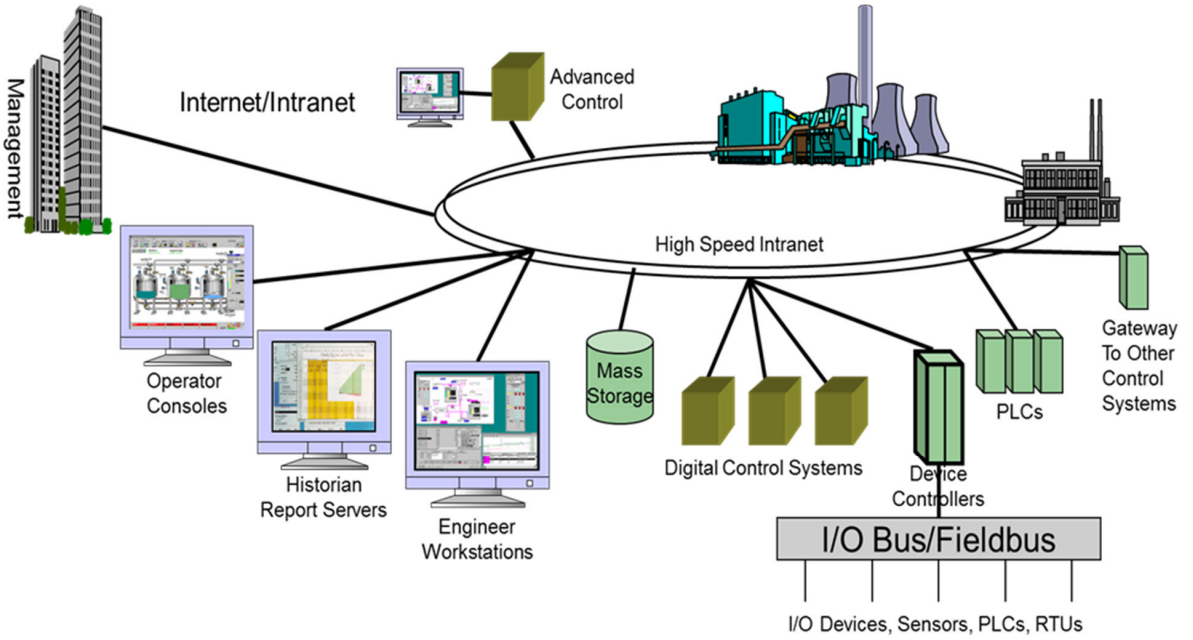

Figure 2 Typical Data Acquisition and Control System Architecture

procedural mitigation to requirements mitigation via technical, non-human means. For example, in the case of the Bhopal failure, the results of an early analysis might have resulted in the inclusion of mechanical interlocks and warnings that pipes were being washed without a slip blind in place. In the case of the MD-80, rather than relying on procedural mitigations to ensure that the jack screw was properly lubricated, requirements for sensors and wear indicators might have been used to warn that proper lubrication was necessary (Hmmm – don't cars have indicators that give a warning when the oil level is low?).

Leveson has proposed the STAMP process (Systems-Theoretic Accident Model and Process) by which the system after completion will be kept in equilibrium via control loops, e.g. feedback on lubrication, interlocks to prevent operator error, etc., rather than relying on human-in-the-loop procedures for safety.[16]

Several years ago, I initiated a research project for a visual language to capture hazards and threats at the very beginning of the systems lifecycle (instead of waiting for draft requirements and architecture). The result of the research was the Unified Requirements Modeling Language (URML), which, during system modeling, treats hazards and threats as first-class concepts.[17]

To summarize then**,** *Risk-centric analysis* is the inclusion of hazard and safety analysis early in the system lifecycle, looking at potential event chains holistically (including the entire operating environment), in addition to relying on experts in narrow areas to identify risks soon after completion of requirements analysis or preliminary design completion. STAMP is a good start, but only if it is applied before the system is designed.

3. Macro and Micro extensibility

The dictionary refers to an extensible system as "a system that can be modified by changing or adding features". I further extend the definition to distinguish between the system as a whole and parts thereof: "macro extensibility" is extensibility of the entire system or product being developed, and "micro extensibility" is the extensibility of individual subsystems or components. (This does not apply to off-the shelf equipment). Researchers such as de Neufville[16] have advocated for flexibility in design, both before and after system completion. However, in this paper, I distinguish between flexibility in design as something that takes place before delivery/installation, and extensibility as something that can be managed by the user or client post-delivery or installation.

For example, a Data Acquisition and Control System (DACS, see Figure 2) is used to control the operation of power plants, refineries, chemical, pharmaceutical plants, etc. Macro extensibility can, for example, include packages that allow the reconfiguration of the entire system to support different domains (e.g. pharmaceutical or power). Often, the time constraints for monitoring equipment are quite different. IEEE 1588 requires sequence of events such as a turbine trip to have an accuracy of +/-100 nanoseconds. However, routine monitoring of plant equipment is typically done at intervals of 10 seconds to 1 minute, depending on the process time constants. Consequently, components that are extensible (i.e. can be configured by the user) to do both sequence of events and routine monitoring have a competitive advantage in the marketplace.

Furthermore, extensibility enables emergent properties. For example, the CETRAN simulator, built in the late 1980s, was originally designed for full scope, replica power plant operation for training plant operators. However, it was sufficiently versatile through its extensible feature set, that it was used for "what-if" studies by the department of energy, and for the design of chemical plants. Thus the CETRAN simulator is an example of a system with emergent properties.[18,19]

4. Error-centric design and construction

When creating system architecture, error prevention can sometimes be an afterthought. For example, when electrical engineers design a circuit board, it is a given that the board will have test points that can be used for determining whether it is operating properly. The test points are used during the manufacturing process, and for field servicing of equipment. Yet when large, complex systems are built, error checking facilities are often added later rather than being designed in. I define error-centric design here as *the inclusion of errors as first class concepts during the initial creation of an architecture and design*, so that the system, subsystems, and components provide test points (for use during manufacturing and servicing), report problems, and, to the extent possible, self-correct and reconfigure (see the paragraph on adaptive systems, below). While resilient systems manage external threats or problems, here I refer to reporting and recovery from internal problems that can have a myriad of causes, e.g. failure due to wear or software bugs, improperly operating equipment or operator actions, failed interconnections, etc.

One typical mechanism for identifying failure is the watchdog timer.[20] The timer, at a prescribed interval, checks its assigned component(s) to ensure proper operation (e.g. "are you ok?"). When the appropriate response is not forthcoming, an error is generated, sometimes in the form of an audible alarm. So error-centric design might include:

- Extensive use of watchdog timers
- Hardware and software test points designed-in
- Wear indicators for hardware (e.g. MD-80 jack screws)
- Appropriate automatic updates for software
- Automated remote diagnostics (i.e. diagnostics that can be initiated remotely, and can identify and repair problems with no human in the loop).

Automated updates are especially conducive to the cultivation of emergent properties; new features, not originally conceived during design, could be added at a later date to provide previously unforeseen benefits.

5. Adaptive and autonomous principles

There is no consensus on the definition of an autonomous system. One definition that may apply when considering the architecture of a system is that of Shani: (paraphrasing) "…a system is recursively self-reliant if it exerts active control over its first-order self-reliance by constructing and maintaining endogenously the boundary conditions which enable it to continue to contribute to its own endurance while facing situations – both external and internal – that would be detrimental otherwise".[21] Most living organisms are autonomous; they can function and maintain their existence without external guidance, especially in the face of hardship. Non-living systems can be considered autonomous if, once they are turned on, they can function and also persevere in the face of hardship without external guidance.

One example of an autonomous system is a space probe such as the Voyager. It can perform its mission in a harsh environment without external guidance. It can recover from power failure, micro-meteor strikes, make decisions and continue to perform its mission (e.g. send back pictures).

A more aggressive, futuristic example of an autonomous system would be a passenger airline, where communication is lost, the pilots are incapacitated, yet the plane can still reach its destination and land without injury to the passengers or damage to the plane.

Holland defines system adaptability as "…how systems can generate procedures enabling them to adjust efficiently to their environments".[22] If we accept that definition, then adaptive systems can be considered a subset of the properties of an autonomous system. A simple example of an adaptive feature is "all-wheel drive", where a vehicle can improve traction based on the road surface by adjusting which wheels drive, and how much power they are given. Adaptive systems, ipso facto, enable emergent properties.

What distinguishes an adaptive system from the banking of technical credit is where:

- Not just the system, but every subsystem or component therein is examined for possible adaptive features
- Adaptive behavior is designed in for longevity, flexibility and reliability rather than to meet a prescribed set of customer requirements.

For example, in the DACS shown in figure 2, every element of the system (excepting connectors) is amenable to some form of adaptive behavior, from switchable power supplies and voltages, to automatically reconfiguring sensors and remote termination devices.

6. Market-centric approach to requirements

The term "market-centric design"**,** sometimes referred to as "user-centric design" is usually applied to "putting people first".[22] That is, users are involved at each stage of the design. I am here referring to something completely different. My alternative definition is: "Using draft marketing material for each software or hardware part (e.g. subsystem, assembly, component…) of the system to better describe the desired end product, whether the part is to be sold independently or not". For example, during the development of a power-plant control system, a middleware communication layer was needed as a platform to support real-time communication between the various components.[18] Heterogeneous computers were used, e.g. even floating point numbers on different nodes used different internal formats. Consequently, the middleware not only managed message flow, but also performed data conversion. In order to better understand and communicate functionality, a product specification sheet was created, as though the intent was to sell the communication layer as a separate product. This approach was then extended to every component of the system, including graphics, I/O, and computation components (some of which included hardware). These specification sheets helped to crystalize for the designers, customers, and management, what the delivered system would include. The specification sheets were distributed at trade shows; the approach worked well enough to increase sales of the CETRAN product. Some of the components were eventually sold to previously unanticipated markets and the graphics and communication subsystems were reused in other commercial power plant products such as thermal analyzers. (These are examples of emergent properties).

Finally, the design and manufacturing of components as part of system development, although heavily supplanted by commercial off-the-shelf components (COTS), is not yet dead. Barry Boehm points out, for example that COTS incompatibilities, upgrades, and loss of support account for a surprisingly large number of system failures.[24] Moreover, there are other reasons for keeping the development of components in house, including improved control of reliability, limited 3rd party warrantees and liability or management of performance characteristics.

7. Over-engineering

Paul Rako, in his editorial on over-engineering states: "Good design is about making compromises on a continuum of choices".[25] That is, over-engineering can be interpreted as simply delivering a system that exceeds the

minimum specified functionality, performance and reliability. I might add "Today's over-engineered system is tomorrow's just-enough system."

One definition of system thinking is "the process of understanding how things influence one another within a whole".[26] Often, during the elicitation of requirements and the design of systems, the question arises, "how much is enough?" The answer to that question can be driven by budget, politics, engineering, uncertainty, or a combination thereof. However, the question is usually asked in the context of a specific short term project, and not in the context of the overall environment (including society). There are many examples of situations where over-engineering "saved the day". The Ponte Vecchio is one example, surviving major floods that destroyed lesser structures.

Over-engineering can be as simple as conscientiously building a system using the best available components and subsystems; it might involve true systems thinking, e.g. who will use this system, how long will it be used, and what happens if it fails?

Over-engineering a system is a form of investment (technical credit) that may pay expected (reputation, longevity of product) or unexpected (emergent properties) dividends. The Ponte Vecchio is just one example. Another example of an over-engineered product is the B-52 bomber. It was introduced in 1952 specifically for delivering nuclear weapons, i.e. as a nuclear deterrent.[27] Now, it is in service as a conventional bomber and is projected to remain in service until 2040 (i.e. note the two emergent properties of unusual longevity and changed role).

8. Conclusions

In this paper, I have suggested that investment in system development, above and beyond the minimum required for product or system delivery, can be considered a form of technical credit. In some cases, this credit can yield dividends many times the cost of the initial investment through emergent properties. I have defined or provided additional definitions for the terms "technical credit", "market-centric design", "error-centric design", macro and micro extensibility, and "risk-centric analysis". Finally, I have suggested a set of heuristics that contribute to the investment in technical credit and may yield unexpected dividends in the form of emergent properties. The use of these heuristics as part of a project checklist (even if considered and discarded) may. Through emergent system properties, may pay unexpected dividends to the project, the customer, and even to society.

Acknowledgement

The author gratefully acknowledges the assistance of Professor Dan Berry of the University of Waterloo for his comments and review of this paper.

References

1. Cunningham, W. "The WyCash Portfolio Management System " in OOPSLA'9 2 Experience report. Vancouver, 1992.
2. Jones, C. *The Economics of Software Quality*, 1st ed. Boston: Pearson Education; 2012
3. Klinger, T., Tarr, P., Wagstrom, P., Williams, C. An Enterprise Perspective on Technical Debt, Proceedings of MTD'11, Honolulu, 2011.
4. http://www.sei.cmu.edu/community/td2013/
5. Toma,E., Auruma,A., Vidgena, R., "An exploration of technical debt", *The Journal of Systems and Software*, Vol 86 pp 1498– 1516, 2013
6. http://technicalcredit.blogspot.com/
7. Melaragno, M., *Preliminary Design of Bridges for Architects and Engineers*. Marcel Dekker.1998 p. 3.
8. http://en.wikipedia.org/wiki/1966_Flood_of_the_Arno_River.
9.Rusch, E.,"Floods destroy or damage more than 50 bridges in Colorado*",* The Denver Post, Denver Sept. 15, 2013.
10. http://www.attractionsworld.net/florence/ponte-vecchio.html.
11. http://en.wikipedia.org/wiki/Vasari_Corridor.
12. Graves, R.J., Goel, A. "Failure Mode Effect Analysis, to Increase Electronic Systems Reliability", 30th International Spring Seminar on Electronics Technology, 2007 , Pp. 128 – 133.
13. Collins, R., de Neufville, R., Claro, J., Oliveira, T., Pacheco, A., "Forest fire management to avoid unintended consequences: A case study of Portugal using system dynamics", Journal of Environmental Management, Vol 130, pp.1-9, 2013.
14. Leveson, N. *Engineering a Safer World*, Cambridge: The MIT Press, 2011.
15. NTSB Number: AAR-02-01, National Transportation Safety Board, December 2002.
16. Leveson, N., Daouk, M., Dulac, N. Marais, K. "Applying STAMP in Accident Analysis", Second Workshop on the Investigation and Reporting of Incidents and Accidents: United States NASA, 2003, pp177-198.
17. Berenbach, B., Schneider, F. "The Use Of A Requirements Modeling Language For Industrial Applications", IEEE 20th International

Requirements Engineering Conference (RE 2012) 2012.
18. Berenbach, B., Koehler, M., Novatin, T. "The Design and Implementation of an Object-Oriented Power Plant Training Simulator", Simulators VIII, SCS Simulation Series, Vol. 24 No. 1 April 1991.
19. Trumpler, D, Alatalo, I. "Implementation of Thermal Hydraulic Network Simulation Programs in CETRAN Environment", VTT SYMPOSIUM, 1994.
20. Pohronská M, Krajčovič T. "Embedded systems with increased reliability using the multiple watchdog timers approach". 2010 International Conference On Applied Electronics (AE 2010) January 2010.
21. Shani, I. ".Setting the bar for cognitive agency: Or, how minimally autonomous can an autonomous agent be?" New Ideas in Psychology, Vol 31 pp151-165 August 2013.
22. Holland, J.H. "Outline for a logical theory of adaptive systems", JACM 3, 297–314 (1962).
23. Roschuni, C., Goodman, E., & Agogino, A. "Communicating actionable user research for human-centered design". AI Edam-Artificial Intelligence For Engineering Design Analysis And Manufacturing, 27(2), 143-154.
24. Boehm, B., Bhuta, J. "Balancing opportunities and risks in component-based software development", *IEEE Software*, 25(6):56-63 2008.
25. Rako,P. Over-engineering: "How much is too much", EDN, Vol. 53 Issue 2, p10, 2008.
26. http://www.bersin.com/lexicon/Details.aspx?id=14310
27. http://en.wikipedia.org/wiki/Boeing_B-52_Stratofortress